ANALYSIS OF SHARED BICYCLE OPERATION

The development of shared bicycles and scooters quickly provides a convenient way for people to travel short distances. It has the characteristics of convenience, low-carbon and environmental protection. However, the extensive operation problems such as indiscriminate parking and unbalanced resource allocation have added a lot of extra maintenance and operation costs to the enterprise and brought a negative impact on the city image. How to fine-tune the operation of shared bicycles and realize dynamic resource allocation has become an important issue. This project uses Spark to preprocess the public data set of shared bikes from Metro Company in Los Angeles area, then analyzes the main reasons for the characteristics of users' riding behaviors and time. Python's sklearn library provides clustering algorithm to analyze and predict the riding demand. With the visualization, the results of the analysis are presented visually.

CCS Concepts: • **Information systems** → **Information storage systems**; **Data management systems**; **Information systems applications**; **Information retrieval**.

Additional Key Words and Phrases: bike-sharing system, urban planning, stations dimensioning, k-means clustering, Spark

ACM Reference Format:

. 2022. ANALYSIS OF SHARED BICYCLE OPERATION. 1, 1 (October 2022), 19 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

1.1 Motivation

Since the twentieth century, much has happened in the nascent field of bike-sharing. There have been three generations of bike-sharing systems over the past 45 years. The first generation was regular bikes of free systems with no fix stations from 1965. The second generation was custom bikes of coin based systems with fix stations from 1995. The third generation was custom bikes of ICT based systems accessed with user card and usually free in the first 30 minutes from 1998 [6]. Nowadays, as long as user has a smartphone and an Internet connection, they can easily use and return bike anytime and anywhere by scanning a QR code on the bike. The shared bicycles becomes a popular way for people to travel short distances. However, fine-grained operation of shared bikes to realize dynamic resource scheduling, improve the turnover rate of bicycles and serve more people with fewer bikes has become an important issue in the development of shared bikes. Nevertheless, data-driven marketing is a new concept that could help salespeople and marketers to understand customers' real needs based on a wealth of data, so that they could acquire more customers and make the best strategy to get the biggest profit. Based on public shared bikes data with specific time and position from different years in Los Angeles, this project's objective is to analyze riding behavior and use K-means method to reduce the complexity among different characteristic types of the data itself, produce realistic results and meet 4V including volume, variety, veracity and velocity.

Author's address:

Manuscript submitted to ACM

1.2 Problem Statement & Challenges

This project aims to build an offline analysis system of shared bicycle riding. With the continuous development of big data technology, there are new methods to analyze user used habits and shared bicycle operation so that find a smart way to plan the shared bicycles parking area, including the Spark distributed computing engine and the Hive data warehouse. One of the challenges in this project is lack of a large amount of data that decrease accuracy. Another challenge is how to adjust K parameter to predict the actual parking conditions. This project hypothesizes that data is enough to show real used behaviors.

2 PRE-PROCESSING

2.1 Data Collection

Metro Bike Share Co., Ltd. provides 210MB of public data for the Los Angeles area from June 2016 to the present, including trip and station details. This project stores the original data as a CSV file every quarter. It can be seen from Figure 1 that there are 15 fields of trip data, which contains the following data points:

1. trip_id: Locally unique integer that identifies the trip.
2. duration: Length of trip in minutes.
3. start_time: The date/time when the trip began, presented in ISO 8601 format in local time.
4. 4.end_time: The date/time when the trip ended, presented in ISO 8601 format in local time.
5. start_station: The station ID where the trip originated (for station name and more information on each station see the Station Table).
6. start_lat: The latitude of the station where the trip originated.
7. start_lon: The longitude of the station where the trip originated.
8. end_station: The station ID where the trip terminated (for station name and more information on each station see the Station Table).
9. end_lat: The latitude of the station where the trip terminated.
10. end_lon: The longitude of the station where the trip terminated.
11. bike_id: Locally unique integer that identifies the bike.
12. plan_duration: The number of days that the plan the passholder is using entitles them to ride; 0 is used for a single ride plan (Walk-up).
13. trip_route_category: "Round Trip" for trips starting and ending at the same station or "One Way" for all other trips.
14. passholder_type: The name of the passholder's plan.
15. bike_type: The kind of bike used on the trip, including standard pedal-powered bikes, electric assist bikes, or smart bikes.

2.2 Data Cleaning

Data preprocessing is the basic part of data analysis. For rich analysis of the solution, the data provided must be clean, accurate, and concise. In fact, the application system to the original data shows the characteristics of incomplete and fuzzy, which is difficult to meet the requirements of data analysis algorithm. Actual data also contains a large number of meaningless dimensions, which not only have a serious impact on the execution efficiency of data mining algorithm, but also cause invalid induction due to noise interference. Data preprocessing has become a key problem in implementing

	trip_id	duration	start_time	end_time	start_station	start_la	start_lon	end_station	end_lat	end_lon	bike_id	plan_duration	trip_route_category	passholder_type	bike_type
1	165226338	19	7/1/2021 0:12	7/1/2021 0:31	4548	34.026829	-118.393517	4555	34.005871	-118.429161	20122	30	One Way	Monthly Pass	standard
2	165226638	22	7/1/2021 0:23	7/1/2021 0:45	4441	34.026619	-118.281807	4441	34.026619	-118.281807	17346	30	Round Trip	Monthly Pass	electric
3	165226838	26	7/1/2021 0:34	7/1/2021 1:00	4555	34.005871	-118.429161	4555	34.005871	-118.429161	14059	30	Round Trip	Monthly Pass	standard
4	165229638	17	7/1/2021 0:50	7/1/2021 1:07	4441	34.026619	-118.281807	4454	34.017899	-118.291718	17346	30	One Way	Monthly Pass	electric
5	165234338	15	7/1/2021 2:38	7/1/2021 2:53	4564	34.035351	-118.434143	4561	34.020061	-118.422684	15665	30	One Way	Monthly Pass	standard
6	165237838	10	7/1/2021 3:25	7/1/2021 3:35	4512	34.043812	-118.264969	4250	34.03017	-118.280952	13872	1	One Way	Walk-up	standard
7	165241038	16	7/1/2021 3:51	7/1/2021 4:07	4217	34.04031	-118.25827	4314	34.057709	-118.279762	19982	30	One Way	Monthly Pass	standard
8	165242038	23	7/1/2021 4:35	7/1/2021 4:58	3014	34.05661	-118.237213	4311	34.059689	-118.294662	18964	30	One Way	Monthly Pass	electric
9	165249038	5	7/1/2021 6:18	7/1/2021 6:23	3074	34.04417	-118.261169	3035	34.048401	-118.260948	12011	1	One Way	Walk-up	standard
10	165249638	6	7/1/2021 6:44	7/1/2021 6:50	4540	34.04776	-118.22541	3079	34.05014	-118.233192	17011	30	One Way	Monthly Pass	electric
11	165252640	22	7/1/2021 6:50	7/1/2021 7:12	4473	34.07159	-118.282288	4245	34.02404	-118.283409	16146	365	One Way	Annual Pass	standard
12	165252438	9	7/1/2021 6:52	7/1/2021 7:01	4511	34.056278	-118.231773	3036	34.039188	-118.232529	20209	30	One Way	Monthly Pass	standard
13	165252638	11	7/1/2021 7:01	7/1/2021 7:12	4300	34.04887	-118.2743	4516	34.045849	-118.254402	13360	30	One Way	Monthly Pass	standard
14	165253042	20	7/1/2021 7:14	7/1/2021 7:34	4212	33.988129	-118.471741	4215	34.014309	-118.491341	6415	30	One Way	Monthly Pass	standard
15	165256251	42	7/1/2021 7:18	7/1/2021 8:00	4254	34.028679	-118.284111	3069	34.05088	-118.248253	13994	1	One Way	Walk-up	standard
16	165253243	23	7/1/2021 7:19	7/1/2021 7:42	3034	34.042061	-118.263382	3014	34.05661	-118.237213	12125	30	One Way	Monthly Pass	standard
17	165253142	15	7/1/2021 7:23	7/1/2021 7:38	3036	34.039188	-118.232529	3027	34.04998	-118.247162	5824	30	One Way	Monthly Pass	standard
18	165253341	18	7/1/2021 7:28	7/1/2021 7:46	4209	33.996239	-118.477448	4214	33.99556	-118.481552	14040	30	One Way	Monthly Pass	standard
19	165253340	18	7/1/2021 7:28	7/1/2021 7:46	4476	34.08252	-118.27272	4303	34.098011	-118.287071	6145	1	One Way	Walk-up	standard
20	165253339	10	7/1/2021 7:36	7/1/2021 7:46	3049	34.056969	-118.253593	4516	34.045849	-118.254402	13903	365	One Way	Annual Pass	standard

Fig. 1. Original Trip Data

data analysis system. Missing values can affect statistical results and lead to analysis bias. Before processing, we use Python functions to look for missing values as Figure 2 and remove them as Figure 3.

```
In [15]: # check missing value
         trip_data_by_year_df.isna().sum()

Out[15]: trip_id             0
         duration            0
         start_station       0
         start_lat           0
         start_lon           0
         end_station         0
         end_lat             0
         end_lon             0
         bike_id             8
         plan_duration       0
         trip_route_type     0
         passholder_type     1
         bike_type           0
         distance            0
         distance_cal        0
         used_date           0
         used_hour           0
         season              0
         holiday             0
         workingday          0
         start_datetime      0
         end_datetime        0
         start_hour          0
         ptd                 0
         dtype: int64
```

Fig. 2. Look for missing value

Based on actual analysis needs, this project drops the two columns of original start and end time. Instead, through data type conversion, the used date with string type, the two columns of start and end datetime with timestamp type and the start hour with string type are added as Figure 4 shows, which make easy for later filtering and aggregation operations. To analyze correlation between riding behavior and riding time, we add one column of distance, which is measured between two places by the latitude and longitude at where they start and end.

In the data preprocessing, trips below 1 minute should be removed and trip lengths are capped at 24 hours. In real life, the cycling distance of citizens less than 100 meters is a small probability event. The reasons may be: 1. The bike itself has quality problems; 2. The wind makes the vehicle fall down, leading to GPS fluctuations; 3. The staff adjusts the position of the surrounding bikes in order to make the bicycles orderly; 4. Users feel that the riding experience is too poor and choose bike by other operating companies. Therefore, data with riding distance less than 100 meters cannot be used as experimental data. This kind of data is eliminated by field distance to obtain better experimental results.

```
In [54]: trip_data_by_year_df = trip_data_by_year_df.dropna(how='any')
         trip_data_by_year_df.isnull().sum()

Out[54]: trip_id            0
         duration           0
         start_station      0
         start_lat          0
         start_lon          0
         end_station        0
         end_lat            0
         end_lon            0
         bike_id            0
         plan_duration      0
         trip_route_type    0
         passholder_type    0
         bike_type          0
         distance           0
         distance_cal       0
         used_date          0
         used_hour          0
         season             0
         holiday            0
         workingday         0
         start_datetime     0
         end_datetime       0
         start_hour         0
         ptd                0
         dtype: int64
```

Fig. 3. Remove missing value

	used_date	start_datetime	end_datetime	start_hour	ptd
1	3/13/2021	2021-03-13T12:35:00.000-08:00	2021-03-13T13:00:00.000-08:00	2021-03-13 12:00:00	2021
2	3/13/2021	2021-03-13T12:35:00.000-08:00	2021-03-13T13:44:00.000-08:00	2021-03-13 12:00:00	2021
3	3/13/2021	2021-03-13T12:35:00.000-08:00	2021-03-13T13:43:00.000-08:00	2021-03-13 12:00:00	2021
4	3/13/2021	2021-03-13T12:36:00.000-08:00	2021-03-13T13:06:00.000-08:00	2021-03-13 12:00:00	2021
5	3/13/2021	2021-03-13T12:37:00.000-08:00	2021-03-13T12:50:00.000-08:00	2021-03-13 12:00:00	2021
6	3/13/2021	2021-03-13T12:39:00.000-08:00	2021-03-13T13:25:00.000-08:00	2021-03-13 12:00:00	2021
7	3/13/2021	2021-03-13T12:39:00.000-08:00	2021-03-13T16:15:00.000-08:00	2021-03-13 12:00:00	2021
8	3/13/2021	2021-03-13T12:40:00.000-08:00	2021-03-13T13:25:00.000-08:00	2021-03-13 12:00:00	2021
9	3/13/2021	2021-03-13T12:40:00.000-08:00	2021-03-13T12:41:00.000-08:00	2021-03-13 12:00:00	2021
10	3/13/2021	2021-03-13T12:41:00.000-08:00	2021-03-13T12:56:00.000-08:00	2021-03-13 12:00:00	2021
11	3/13/2021	2021-03-13T12:41:00.000-08:00	2021-03-13T16:15:00.000-08:00	2021-03-13 12:00:00	2021
12	3/13/2021	2021-03-13T12:42:00.000-08:00	2021-03-13T13:25:00.000-08:00	2021-03-13 12:00:00	2021
13	3/13/2021	2021-03-13T12:42:00.000-08:00	2021-03-13T13:06:00.000-08:00	2021-03-13 12:00:00	2021
14	3/13/2021	2021-03-13T12:43:00.000-08:00	2021-03-13T13:25:00.000-08:00	2021-03-13 12:00:00	2021
15	3/13/2021	2021-03-13T12:46:00.000-08:00	2021-03-13T13:14:00.000-08:00	2021-03-13 12:00:00	2021
16	3/13/2021	2021-03-13T12:48:00.000-08:00	2021-03-13T13:08:00.000-08:00	2021-03-13 12:00:00	2021
17	3/13/2021	2021-03-13T12:49:00.000-08:00	2021-03-13T13:55:00.000-08:00	2021-03-13 12:00:00	2021
18	3/13/2021	2021-03-13T12:49:00.000-08:00	2021-03-13T13:47:00.000-08:00	2021-03-13 12:00:00	2021
19	3/13/2021	2021-03-13T12:52:00.000-08:00	2021-03-13T12:53:00.000-08:00	2021-03-13 12:00:00	2021
20	3/13/2021	2021-03-13T12:53:00.000-08:00	2021-03-13T13:18:00.000-08:00	2021-03-13 12:00:00	2021

Fig. 4. Add New Columns Related To Time

2.3 Feature Engineering

Data is the carrier of information, but the original data contains a lot of noise, and the expression of information is not concise. Therefore, the purpose of feature engineering is to use a more efficient coding method (feature) to represent such information through a series of engineering activities. This project looks for outliers based on descriptive statistics as Figure 5, and the descriptive statistics shows that the data is within the normal range, and the minimum and maximum value are reasonable except for longitude and latitude. The range of longitude and latitude extends beyond the latitude and longitude of the Los Angeles metropolitan area studied for this project. By getting geography shape with MultiPolygon type and json format from Opendatasoft website as Figure 6 shows, this project filter out the latitude and longitude that fit the study area by python package of shapely.geometry. This step ensures the authenticity of data and accuracy of research results.

To realize business digitalization and save storage space, based on feature transformation that converts text type to numeric type, this project uses the PySpark UDF operation to add three columns of season, holiday, and workingday data with Integer type to save storage space as Figure 7 shows. For the season, 1 represents winter; 2 represents spring;

```
In [41]: ptd = partitions_list.collect()
         desc_list_names = ['duration', 'start_lat', 'start_lon', 'end_lat', 'end_lon', 'plan_duration',  'trip_route_type',
                            'passholder_type', 'bike_type', 'distance', 'season', 'holiday', 'workingday']


         for year in ptd:
             cur_year = year[0]
             print(cur_year)
             df_by_year = spark.sql('select {} from trip_details where {};'.format(','.join(desc_list_names), cur_year))
             df_by_year.toPandas().describe().T

         ptd=2020
```

Out[41]:

	count	mean	std	min	25%	50%	75%	max
duration	126172.0	40.957146	112.461548	1.000000	9.000000	18.000000	31.000000	1440.000000
start_lat	126172.0	34.096126	1.021466	33.928459	34.039982	34.048401	34.059689	55.705528
start_lon	126172.0	-117.947782	7.349526	-118.491341	-118.296593	-118.263382	-118.253838	37.606541
end_lat	126172.0	34.094481	1.006877	33.928459	34.039982	34.048401	34.058960	55.705528
end_lon	126172.0	-117.957867	7.244404	-118.491341	-118.296799	-118.263382	-118.253838	37.606541
plan_duration	126172.0	49.083838	96.996505	1.000000	1.000000	30.000000	30.000000	999.000000
trip_route_type	126172.0	1.249485	0.432717	1.000000	1.000000	1.000000	1.000000	2.000000
passholder_type	121559.0	2.526436	0.929749	1.000000	2.000000	3.000000	3.000000	4.000000
bike_type	126172.0	1.575167	0.666525	1.000000	1.000000	1.000000	2.000000	3.000000
distance	126172.0	457.494263	11475.629883	0.000000	2.118922	16.547836	33.141544	304154.843750
season	126172.0	2.854508	0.876839	1.000000	2.000000	3.000000	4.000000	4.000000
holiday	126172.0	0.027748	0.164250	0.000000	0.000000	0.000000	0.000000	1.000000
workingday	126172.0	0.663832	0.472399	0.000000	0.000000	1.000000	1.000000	1.000000

Fig. 5. Outliers Check

STUSAB	Geo Point	Geo Shape	STATEFP	COUNTYFP	COUNTYNS	GEOID	NAME
CA	34.1979978627, -118.26101385	{"type":"MultiPolygon","coordinat...	06	037	00277283	06037	Los Angeles

Fig. 6. Area Shown by MultiPolygon

3 represents summer; 4 represents autumn. For the holiday, 1 represents holiday; 0 represents non-holiday. For the workingday, 1 represents working day; 0 represents holiday or weekend.

	trip_id	start_datetime	season	holiday	workingday
1	155146002	2021-03-13T12:35:00.000-08:00	1	0	0
2	155146874	2021-03-13T12:35:00.000-08:00	1	0	0
3	155146873	2021-03-13T12:35:00.000-08:00	1	0	0
4	155146105	2021-03-13T12:36:00.000-08:00	1	0	0
5	155144191	2021-03-13T12:37:00.000-08:00	1	0	0
6	155146426	2021-03-13T12:39:00.000-08:00	1	0	0
7	155172034	2021-03-13T12:39:00.000-08:00	1	0	0
8	155146425	2021-03-13T12:40:00.000-08:00	1	0	0
9	155144088	2021-03-13T12:40:00.000-08:00	1	0	0
10	155144393	2021-03-13T12:41:00.000-08:00	1	0	0

Fig. 7. Business Digitalization

Dummy Variables are artificial Variables used to reflect qualitative attributes. They are quantified independent Variables, usually with the value of 0 or 1. The introduction of dummy variables can make linear regression model more complex, but the problem description is more concise, one equation can achieve the function of two equations, and close to reality. This project separates four columns of plan duration, trip route type, passholder type, bike type and time period type into dummy variables as Figure 8. This form of storage is beneficial to correlation analysis among different attributes, because each attribute is independent.

trip_id	start_hour	PLAN_DURATION_DAY	PLAN_DURATION_MONTH	PLAN_DURATION_YEAR	TRIP_ROUTE_TYPE_ONE_WAY	TRIP_ROUTE_TYPE_ROUND_TRIP	PASSHOLDER_TYPE_WALK_UP	PASSHOLDER
155146002	2021-03-13 12:00:00	1	0	0	1	0	1	
155146874	2021-03-13 12:00:00	1	0	0	1	0	1	
155146873	2021-03-13 12:00:00	1	0	0	1	0	1	
155146105	2021-03-13 12:00:00	1	0	0	1	0	0	
155144191	2021-03-13 12:00:00	1	0	0	1	0	0	
155146426	2021-03-13 12:00:00	1	0	0	1	0	1	
155172034	2021-03-13 12:00:00	1	0	0	1	0	0	
155146425	2021-03-13 12:00:00	1	0	0	1	0	1	
155144088	2021-03-13 12:00:00	0	0	1	0	1	0	
155144393	2021-03-13 12:00:00	0	0	1	1	0	0	
155172030	2021-03-13 12:00:00	1	0	0	0	1	0	
155146543	2021-03-13 12:00:00	1	0	0	1	0	1	
155146099	2021-03-13 12:00:00	0	1	0	1	0	0	
155146539	2021-03-13 12:00:00	1	0	0	1	0	1	
155146200	2021-03-13 12:00:00	0	1	0	1	0	0	
155146096	2021-03-13 12:00:00	0	1	0	1	0	0	
155147252	2021-03-13 12:00:00	1	0	0	1	0	1	
155146960	2021-03-13 12:00:00	1	0	0	1	0	1	
155144288	2021-03-13 12:00:00	0	1	0	1	0	0	

Fig. 8. Dummy Variables

3 ANALYTICAL FRAMEWORK

3.1 Feature analysis

3.1.1 Aggregate statistics. The Spark SQL aggregation function is used to calculate the number of records in an hour, a day, workday of a week, weekend of a week, a month, and a season. The data as Figure 9 shows that the total number of working days is less and there are more idle bicycles. On weekends, some citizens like outdoor sports, resulting in increased demand for bicycles. The standard bikes are used more often than electric bikes, probably because of the higher cost of riding. The number of users with One Day Pass was the lowest, and the number of users with Monthly Pass was higher than that of walk-up, which means high user retention rate. Monthly Pass is more favored than One Day Pass by consumers and has a high utilization rate of riding. Therefore, it is necessary to increase the market share of this Pass. For different seasons, the number of bicycles is a bit more frequent in the summer and autumn, while the number of bicycles is relatively small in winter due to the influence of climate and temperature. According to the above analysis, the time distribution of the number of bicycle rides is closely related to the daily activities of citizens.

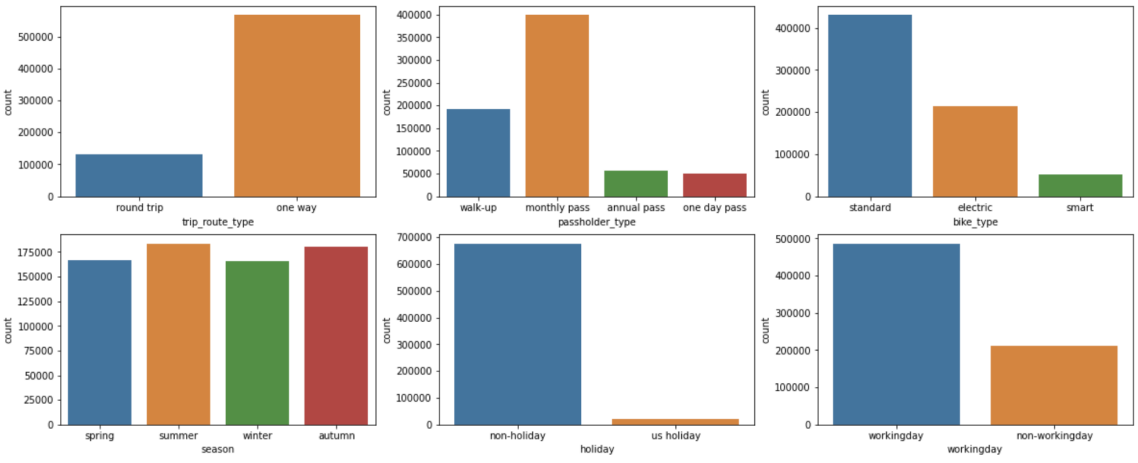

Fig. 9. Feature Analysis in Recent Three Year

3.1.2 Pointplot in categorical variable. The point plot represents the central trend estimate of the numerical variable at the position of the scatter plot and uses error lines to provide some indication of the uncertainty of that estimate. In

this project, as can be seen from the figure 10, the time of day and whether it is a working day have a greater impact on people's willingness to ride. For example, people prefer to ride on non-working days. However, season and holidays have a greater fluctuation on people's cycling intentions, and the two-level differentiation is more obvious.

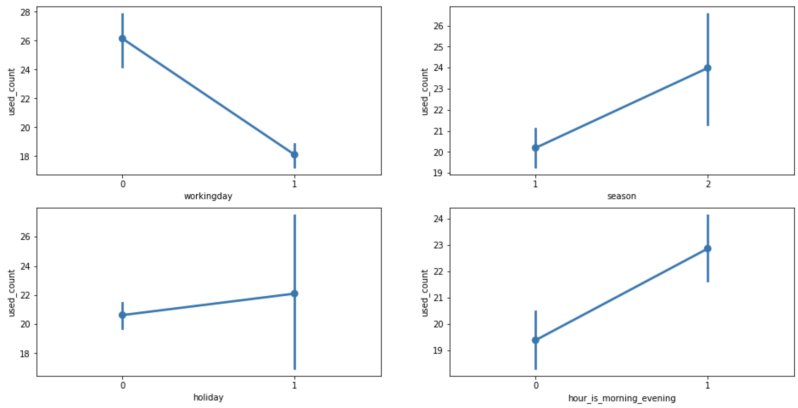

Fig. 10. Fluctuation Reflected by Pointplot in 2021

3.1.3 Correlation coefficient. The correlation matrix is also called the correlation coefficient matrix, which is composed of the correlation coefficients between the columns of the matrix. The coefficient is a statistic that reflects the degree of linear correlation between two variables. In order to compute the correlation coefficient, the following formula is applied:

$$\rho(X, Y) = \frac{cov(X, Y)}{\sigma_X \sigma_Y} \tag{1}$$

Cov(X, Y) is the covariance, and the two variables in the denominator represent the standard deviation of X and Y. The correlation coefficient is at [-1, 1], and the closer it is to 1 or -1, the stronger the positive/negative linear relationship is. 0 indicates that there is no linear relationship between the two variables. This project uses heatmap to show correlation matrix as Figure 11. A statistical indicator reflecting the linear correlation between two variables. The correlation matrix shows that there was a strong correlation between standard bike use and one-way and day pass users, which means that day pass holders were more likely to use standard bikes, while standard bike users were more likely to use one-way bikes. There is a strong correlation between the use of electric bikes and the monthly card holders, which means that the monthly card holders are more inclined to use electric bikes. The correlation between smart bikes and user types is not obvious. To increase the usage of smart bikes, marketers should adjust the price of smart bikes.

3.2 Modeling analysis

PySpark framework provides many common machine learning algorithms for developers in MLlib library, which mainly include regression, binary classification, clustering, collaborative filtering and gradient descent optimization. K-means algorithm is a clustering algorithm that can group data points, and it is an effective clustering algorithm based on partition, which has been widely used for clustering analysis [9]. The K-means of MLlib provide two random initialization methods, each model training results are different. K-means usually do not converge to the global optimal solution, so it is necessary to train and select the optimal model in practical application. After the loss function is

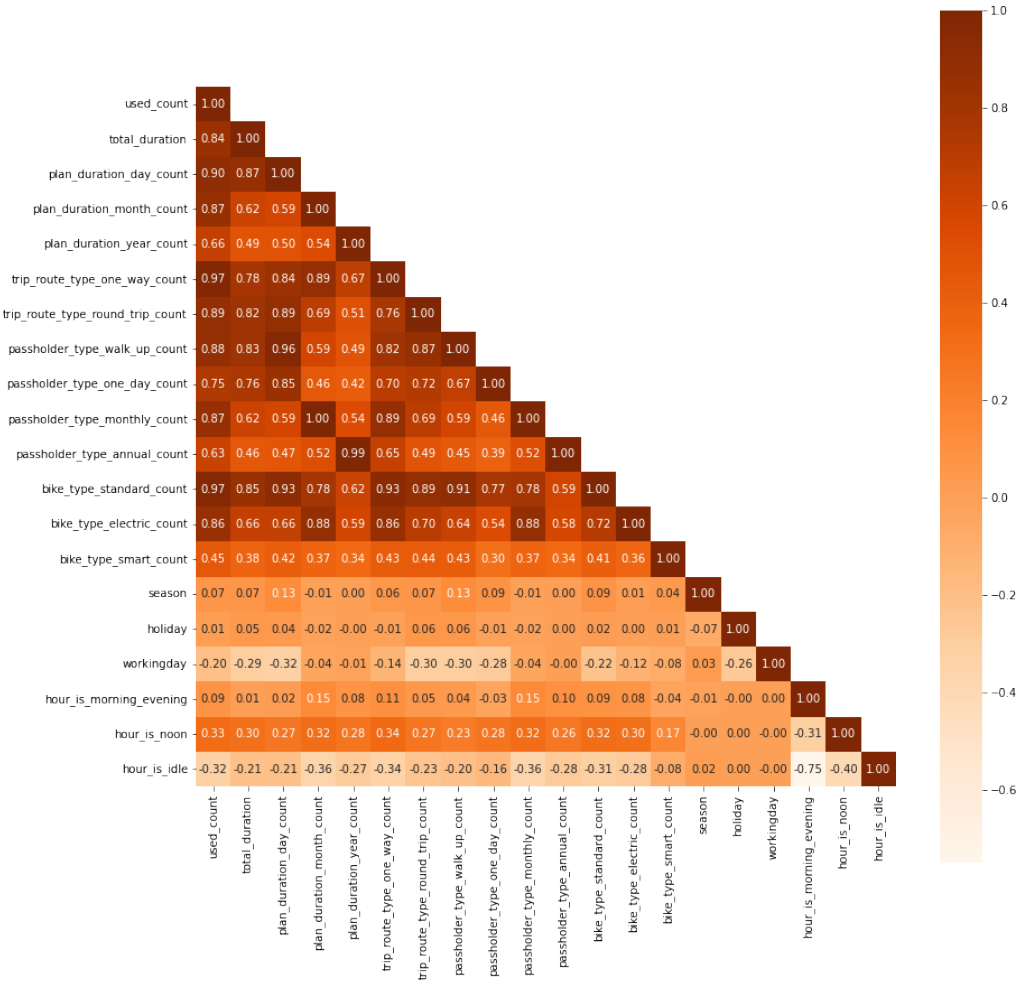

Fig. 11. Heatmap of Correlation Matrix in Recent Three Year

evaluated, the best performance evaluation is selected as the final model. Next, the trained K-means model was used to cluster the data in the test set to generate the prediction results, and the clustering results were further analyzed. First, the transform() of the clustering model is called to cluster the test data, and the predicted result is of type DataFrame. The running time of Spark is proportional to the value of K, that is, the larger the value of K is, the longer the running time of Spark is. Better performance can be achieved as the number of iterations increases, but performance increases more slowly as the number of iterations reaches a certain level.

This project first calculates the neighborhood distance threshold required for eps parameter of DBSCAN through NearestNeighbors, then builds clustering model to get the cluster center. Through the statistical analysis of the prediction results of the clustering of the borrowing and returning locations of bicycles, the demand situation of high frequency starting point and high frequency ending point is obtained. Figure 12 shows high frequency starting station of riding

demand. According to the prediction results of clustering, marketers can plan parking locations and set up recommended parking spots in accordance with the demand of cluster points in all starting locations.

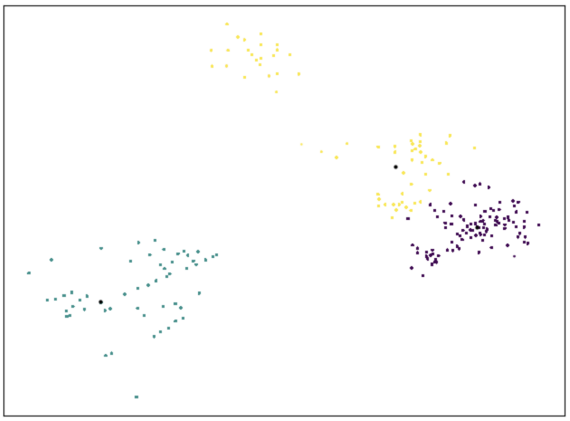

Fig. 12. High Frequency Starting Station Clustering

4 SYSTEM DESIGN

4.1 Data Warehouse Architecture

The data warehouse is a topic-oriented, integrated, time-changing collection of data, and the information itself is relatively stable. A data warehouse is designed to layer data so that it is easy to locate and understand when using tables. Standardizing data layering and developing some common middle tier data can reduce significant double calculations. Breaking a complex task down into multiple steps, with each layer dealing with a single step, is relatively simple and easy to understand. In addition, it is convenient to maintain the accuracy of data. When there is a problem with data, it is not necessary to repair all the data, but only need to repair from the steps with problems. Based on the advantages of data warehouse, this project choose Hive to build a three-tier data warehouse model to store data, including operational data store tier, data warehouse tier and application data store tier. In operational data store (ODS) tier, we store the original data as csv format. In data warehouse detail (DW) tier, dimension degradation method is adopted to reduce the association between fact table and dimension table at ODS tier, and wide table is used to construct common indicator data layer to improve the ease of use of detailed table. Use parquet column storage to improve query efficiency because of higher compression ratio and smaller I/O operations. In application data store (ADS) tier stores personalized statistical index data, which is mainly for front-end analysis and virtualization. Because this project uses offline data, full data updates are used instead of incremental data updates. Partitions in Hive divide table data roughly based on partition column values. In Hive, a table corresponds to a directory, and subdirectories are created in this table directory based on partition columns. Each subdirectory name is the name of the partition column. A partitioned column definition is similar to a field in a table, but is independent of a field in a table. This speeds up the data query because the entire table is not scanned. This project adds ptd field of year to partition data according to time series to improve the speed of data query, and the storage directory is shown as Figure 13.

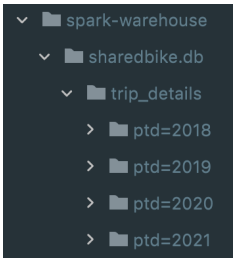

Fig. 13. Partition By Year

4.2 System Application

This project uses Python, PySparkSQL and Hive to realize analysis of shared bicycle operation. Because of the short development cycle and simple syntax, this project chose Python as the programming language of development. This project utilizes PySparkSQL to clean and perform transformation of data, Python's scipy library to analyze the data, Python's sklearn and PyMllib to build model, and Python's matplotlib and seaborn to visualize data. This project choose Hive to store data, because it is not only easy to operate and configure by PySpark, but also features a data warehouse that clearly divides service layers and supports user-defined functions.

4.3 Visualization

To present our data, analysis results are presented primarily through Jupiter in the following ways.

4.3.1 Plot. On the one hand, the chart is convenient for users to understand a large amount of data and the relationship between data. On the other hand, the chart allows users to read the original data more quickly and intuitively through visual symbols.

4.3.2 Geographic Visualization. Figure 14 gives an overview of geographic distribution of the starting latitude and longitude, which obviously shows the utilization density area. Almost all the stations are located in Downtown, Santa Monica and East Hollywood with the exception of one station, whose id is 4403. The 4403 station was only used for a week from data, probably because it was in operation for a week.

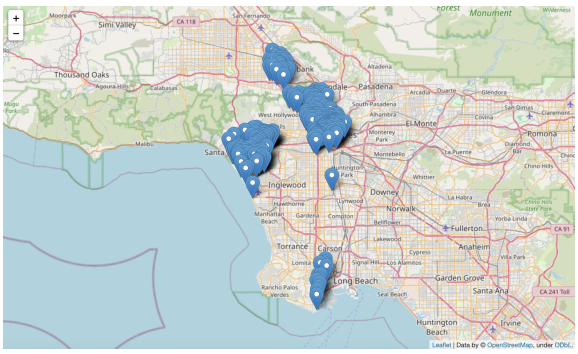

Fig. 14. Map of Start Station Distribution

4.3.3 Heatmap. Heat maps can show the differences of data, especially for huge data, and can intuitively understand the distribution or differences of data through heat map visualization. This project uses heatmap to show the correlation matrix, which is composed of the correlation coefficients between the columns of the matrix and can intuitively show the correlation between various factors.

5 EVALUATION

5.1 Performance Analysis

5.1.1 Spark Explain Plan Log. Spark Explain is used to provide logical/physical plans for an input statement. By default, this clause provides information about a physical plan only. This project prints explain of dataframe and shows the optimized suggestion. Figure 15 gives an example of explain when generating a dataframe of trips count hourly. The optimized logical plan suggests that we should use batcheval function instead of projection and add isnotnull() into filter to improve performance. In this way, Sparksql performance can be accurately evaluated and recommended for optimization.

Fig. 15. Pyspark Explain Output

5.1.2 Spark UI. When a Spark Application is running, you can access the UI by accessing the hostname:4040 port. After the Spark task is submitted, the tracking URL, which is the log link of the task, is displayed in logs. After tracking URL is opened in the browser, the Jobs, Stages, Storage, Environment, Executors, and SQL tabs are displayed.

1. Jobs tab. Jobs consists of two parts: Event Timeline, which refers to the Timeline information of events; The current application analyzes all tasks, including the execution times of all executors actions, etc. This shows all jobs in Active, Completed, Canceled, and Failed states as Figure 16.
   The Event Timeline graph shows the point in time that executor created, the operator tasks triggered by an action, and when they were executed as Figure 17.

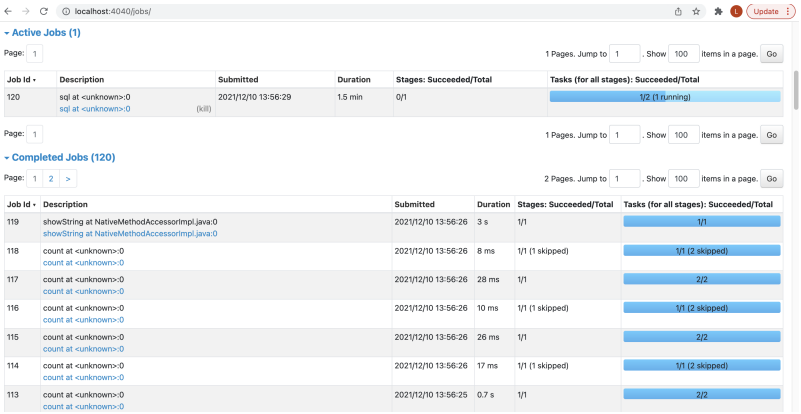

Fig. 16. Spark Jobs with Different States

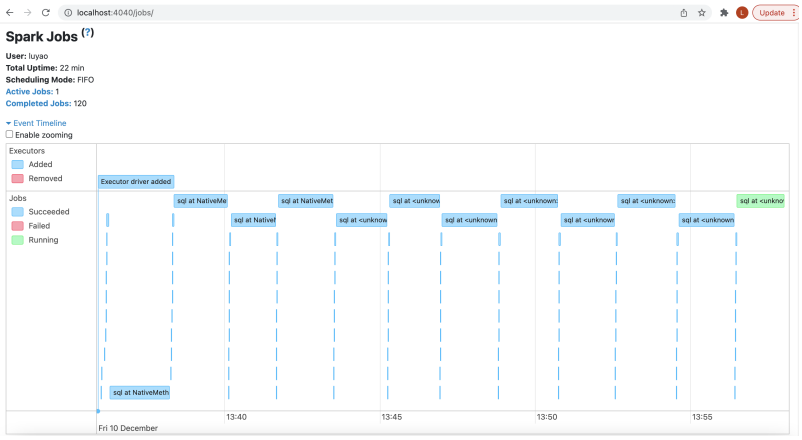

Fig. 17. Spark Jobs Event Timeline

Using this time chart, you can quickly find the execution bottleneck of the application and how many actions are triggered. Duration shows the Duration of the action, which can also be used to specifically optimize the code. The final progress bar shows the number of failures and successes of the task. If there are any failures, specific analysis is required.

2. Stages. On the Stages page, you can view all Stages of an application as Figure 18. Stages are classified by width dependency and are finer in granularity than Jobs. Click on the link in Description to see the stage page for the action, including details such as the DAG diagram as Figure 19, and execution information for executor and task. DAG diagrams mark a flow chart for each RDD from creation to application and are important for analysis and tuning.
3. Storage. Storage provides information about operations such as cache persist that have been performed, how much cache is currently in use, and clicking on a specific RDD to see which blocks are being used on each machine as Figure 20.

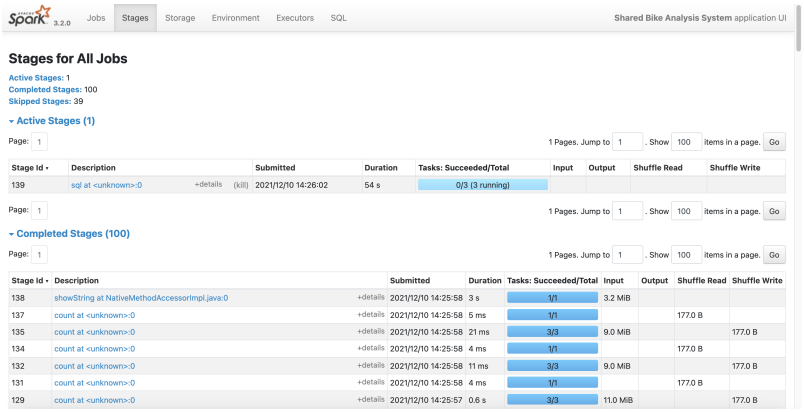

Fig. 18. Spark Stages

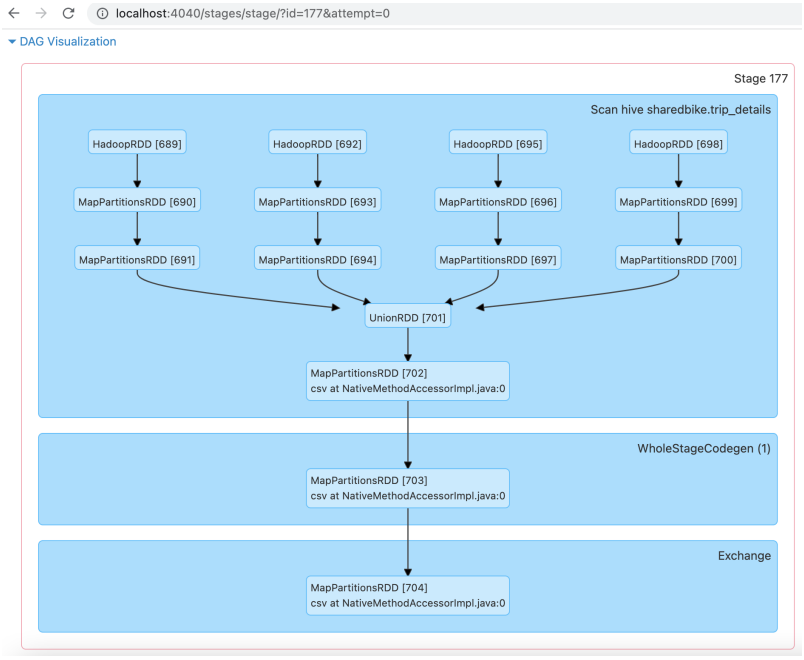

Fig. 19. Spark Stages DAG

4. Executor. The Executors provides information about memory, CPU cores, and other resources used by Executors as Figure 21. This information is available at both the Executor and summary levels. On the one hand, it can be seen whether data skewness occurs in each executor; on the other hand, it can be specifically analyzed whether the current application generates a large number of shuffle, and whether the data volume of shuffle can be reduced by data localization or reducing data transmission.

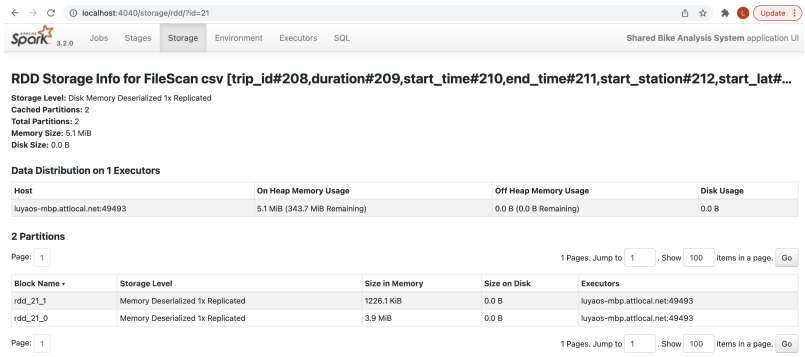

Fig. 20. Cache Information from Spark Storage

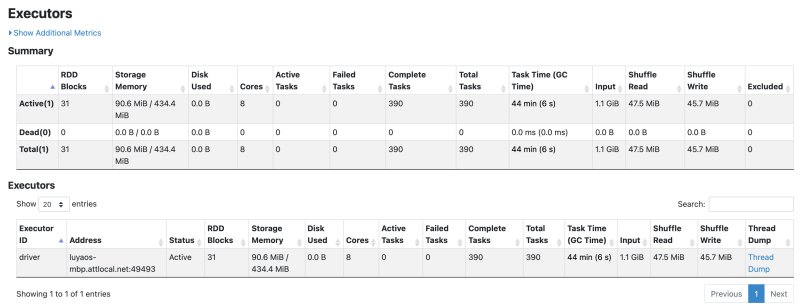

Fig. 21. Spark Executor Information

5. SQL. The SQL TAB allows you to view the details of the SQL execution plan, which provides a DAG for SQL queries and a query plan that shows how Spark optimizes executed SQL queries as Figure 22.

5.1.3 Coverage. Coverage is to check the code coverage, check the logic coverage of the back-end business logic interface, so as to achieve a comprehensive test. According to the need to cover important business code, so as to provide the comprehensiveness of test cases and improve the test quality. The code coverage of this project is 52% as Figure 23, which is relatively low due to the existence of uncalled utility class methods in common utility classes, while the coverage of business class code is relatively high.

5.2 Partition and cache optimization

PartitionBy function can greatly speed up queries by partitioning data on disk. Before partitioning the data by year, this project uses the repartition() function to prepartition the data in memory before writing it to disk. Because the data volume is less than 300MB, too many partitions will cause the problem of reading small files and reduce the read and write efficiency, this project only need to set the number of files in each partition to 1 as Figure 24. If the amount of data is too large, we need set the number of files in each partition properly and use maxRecordsPerFile to adjust the number of small files. In addition, adjusting the maximum number of files in a partition based on the file size and

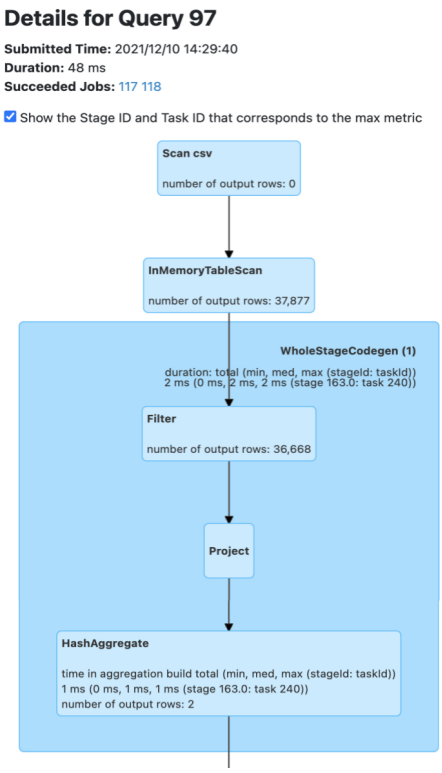

Fig. 22. Spark SQL Query Plan

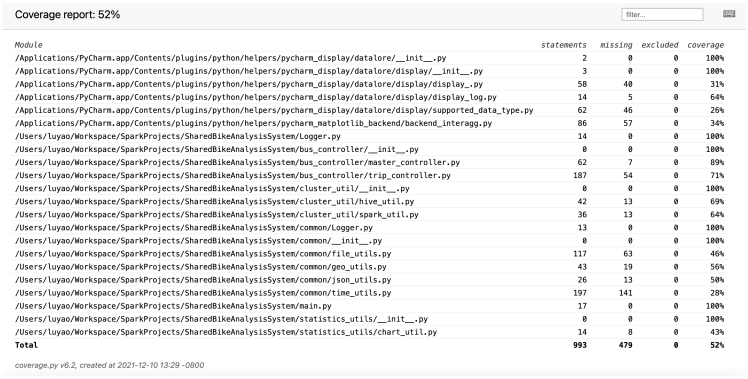

Fig. 23. Coverage Report

number could avoid hot issues. To calculate the maximum number of output files, the following formula is applied:

$$Maximum\ number\ of\ output\ files = Number\ of\ partitions\ in\ memory\ x\ Number\ of\ different\ partitions = repartition\ x\ partitionBy \quad (2)$$

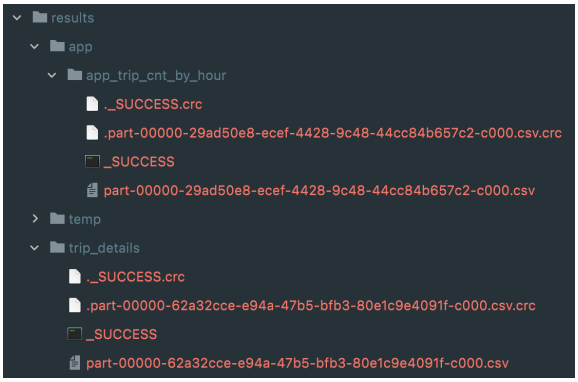

Fig. 24. Partition Directory Structure

Spark SQL can cache tables using an in-memory columnar format by calling spark.catalog.cacheTable("tableName") or dataFrame.cache(). Then Spark SQL will scan only required columns and will automatically tune compression to minimize memory usage and GC pressure.

5.3 K-means clustering evaluation

Clustering is an unsupervised learning algorithm, in which the mark of the training sample is unknown. According to the inherent nature and law of a certain standard or data, the sample is divided into several disjoint subsets. Each subset is called a cluster, and each cluster contains at least one object, and each object belongs to and only belongs to one cluster. The data similarity within clusters is high, but the data similarity between clusters is low. Clustering can be used as a precursor to other learning tasks such as classification. This project uses two evaluation methods: Davies-Bouldin index(DBI) and silhouette coefficient.

5.3.1 DBI evaluation. DBI, also known as classification accuracy index, is an index to evaluate the advantages and disadvantages of clustering algorithms. DBI is used to calculate the ratio of the sum of the distances within the class to the distances outside the class, so as to optimize the selection of k value and avoid the local optimal situation caused by only calculating the objective function in k-means algorithm. First suppose we have m time series that are clustered into n clusters. M time series were set as input matrix X, and N cluster classes were set as N as parameters passed into the algorithm. Use the following formula to calculate:

$$DBI = \frac{1}{N}\sum_{i=1}^{N}\max_{j\neq i}\left(\frac{\bar{S}_i + \bar{S}_j}{||\omega_i - \omega_j||_2}\right) \tag{3}$$

The meaning of this formula is to measure the mean of the maximum similarity of each cluster class. The minimum value of DBI is 0. The smaller the value is, the better the clustering effect is. In this project, the DBI of the clustering model calculated is 1.43.

5.3.2 Silhouette coefficient evaluation. Silhouette Coefficient is a means of evaluating the clustering effect. It combines the factors of cohesion and separation. It can be used to evaluate the influence of different algorithms or different operating modes of algorithms on clustering results on the basis of the same original data. The value of the contour coefficient of the clustering result is between [-1,1]. The larger the value is, the similar samples are about close to each

other; the farther the different samples are, the better the clustering effect is. To define the contour coefficient of sample is zero is:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{4}$$

In this project, the Silhouette Coefficient of the clustering model calculated is 0.648, a large value closed to 1, which means the similar samples are close to each other, the apart different samples are far and the clustering effect is better.

5.4 Data authenticity

Through the SHAPE file of Los Angeles area as Figure 25, this project verifies whether the latitude and longitude of the experimental data is within the scope of this city to ensure the authenticity of the data. The standard deviation, maximum value and minimum value are used to measure whether the data of each dimension is within the normal range.

Fig. 25. Range of Los Angeles

6 RELATED RESEARCH

In this section, we summarize the related works in three main areas: 1) statistics of bicycle data based on high-dimensionality, 2) prediction model in specific zone, and 3) urban planning.

6.1 Statistics of Bicycle Data Based on High-Dimensionality

The majority of the quantitative analysis focuses on state prediction using time series models. The high-dimensionality of the data involved calls for a dimensionality reduction [8]. Based on time analysis, the most important activities of the stations are characterized by 3 peaks every ordinary day. For different seasons, holidays and weekends, the peaks of activity are easily identified. Time patterns of flows between stations display similarities so that they are grouped in clusters separating trips related to professional activities (weekdays and major communication hubs) from those used during leisure time (weekend and parks). Depending on the time in the week, some stations are alternatively

sinks [3]. In addition, the recent studies examined the effects of transportation infrastructure, bike-sharing facilities, land-use and built environment characteristics, and temporal characteristics on bike- sharing usage, such as trip distance, temperature, precipitation, and poor air quality [5]. The overall trend is that the worse the weather, the less cycling is needed. Other specific indicators of weather are temperature, body sense temperature, humidity and wind speed. According to common sense, there should be a certain correlation between temperature and body sense temperature, humidity and wind speed.

6.2 Prediction Model in Specific Zone

Another related stream of literature focuses on the use of data mining methods, such as clustering and prediction, which is helpful to optimize bike repositioning operations. The model result should show the relationship between the usage of bike-sharing was affected by gender, trip model, travel time, bike-sharing stations location, and users' perception of bike-sharing [10]. The distances between every couples of pairs of stations were evaluated classically by the correlation between the temporal vectors of number of rentals [2]. To identify the probability that a person from a given cluster decides to use the system, three algorithmic options, logistic regression, decision trees (XGBoost), and neural network could provide prediction result [4].

6.3 Urban Planning

Finally, literature focuses on practical application of bike sharing analysis. Constructing well-planned bike paths and parking zone is not only beneficial to retain more users and keep profit, but also effective to reduce traffic congestion and decrease safety risks. According to the existing road conditions and cycling paths, dynamic analysis of bike lane demand can effectively improve the efficiency of urban operation, improve the living standards of urban residents, and build a smart city. However, based on our investigation, most of the government current strategy, building bike lanes only on major roads is insufficient, and the cycling conditions need to be improved in the local road segments. The analysis of large-scale data from users' riding routes make smart bike lane planning possible. With the shared bicycle system usage increased, there are more bicycle facilities, such as bicycle lanes and bicycle paths, near a bike-sharing station [1]. Based on large-scale real data of shared bicycles, the data-driven approach could make developing bike lane construction plans possible. To enforce constraints of construction budget constraint, connectivity constraint and maximum usage benefit, the greedy-based heuristics with the top-k based and spatial clustering is an effective approach to solve the NP-hardness problem. The top-k guarantees that the algorithm will never miss any segment with the highest beneficial score per cost. In order to include more spatially diversified starting locations in the initialization stage and be more effective when the budget is larger, spatial clustering techniques provide an approach to select the starting road segments [7].

7 CONCLUSION

In conclusion, based on the analysis of user usage data, more shared bikes should be placed at frequently used times and places, and prices of different types of bikes and members should be adjusted according to user usage. We released the source code of our prototype and analysis to facilitate the reproduction of results and future research: https://github.com/AstroMen/SharedBikeAnalysisSystem.git.

ACKNOWLEDGMENTS

We thank Metro Bike Share Co., Ltd. for providing the trip data with the public.

REFERENCES

[1] El-Geneidy A Bachand-Marleau J, Lee B. 2012. Better understanding of factors influencing likelihood of using shared bicycle systems and frequency of use. *Transportation Research Record: Journal of the Trans-portation Research Board* 2314 (2012), 66–71.

[2] M. Basseville. 1989. Distance measures for signal processing and pattern recognition. *Signal Processing* 18, 4 (1989), 349–369.

[3] et al Borgnat, Pierre. Febuary 2010. Shared Bicycles in a City: A Signal Processing and Data Analysis Perspective. *Scientific Commons* (Febuary 2010).

[4] Hadi Feyzollahi Cai Gao, Ningji Wei and Luca Wrabetz. 2018. Improving the Service Quality of Bike Sharing Systems via the Analysis of Real-Time User Data. (2018).

[5] Ryerson MS-Yang X Campbell AA, Cherry CR. 2016. Factors influencing the choice of shared bicycles and shared electric bikes in Beijing. *Transportation Research Part C: Emerging Technologies* 67 (2016), 399–414.

[6] P. Will DeMaio. 2004. smart bikes succeed as public transportation in the United States? *Journal of Public Transportation* 7, 2 (2004), 1–15.

[7] Sijie Ruan-Yanhua Li Yu Zheng Jie Bao, Tianfu He. August 2017. Planning Bike Lanes based on Sharing-Bikes' Trajectories. *KDD 2017* (August 2017).

[8] Lavanya Marla R. Hampshire. 2012. An Analysis of Bike Sharing Usage: Explaining Trip Generation and Attraction from Observed Demand. *TRB 2012 Annual Meeting* (2012).

[9] Li Xiangyu Xiao Junwei, Lu Jianfeng. 2017. Davies Bouldin Index based hierarchical initialization K-means. *Intelligent Data Analysis* 21, 6 (2017), 1327–1338.

[10] Yao Wu-Zhibin Li Yanyong Guo, Jibiao Zhou. 2017. Identifying the factors affecting bike-sharing usage and degree of satisfaction in Ningbo, China. PLoS ONE. *Signal Processing* 12, 9 (2017), e0185100.